Notes on using R programming in Survival Models

Chris Sutton, October 2021

1. Downloading the software

You should have done this already for previous Statistics modules at QMUL but just in case.

For R itself go to https://www.r-project.org/ then select CRAN from the left hand side and go to one of the sites in the country you are based in to download the latest version of the software in the operating system you are using (e.g. https://cran.ma.imperial.ac.uk/ in the UK).

You will then need R-Studio which you can download from https://www.rstudio.com/products/rstudio/download/ in order to run calculations in R.

When you open R Studio the panel on the left hand side is the Console where you write R code. In the top right panel there are a number of tabs. The one labelled Environment in sometimes called the Workspace where values of variables and data sets are displayed. The lower right panel is where files are shown, as well as Plots which is a graphics window, and a Packages tab which we will come to later.

2. Basic mathematics in R

R does things when we type commands in the Console.

basic arithmetic uses +, -, *, / for addition, subtraction, multiplication and division so e.g. typing 3+6 in the console will return [1] 9 as the result and 4/9 in the console will return [1] 0.4444444. Using %/% will give the integer part of the division so 38 %/% 5 returns [1] 7. You raise to the power by either ^ or **, so 3^4 and 3**4 both return [1] 81. The [1] index is common to command returns in the R console.

Typing CTRL L will clear the Console of all code.

If you need help within R type help.start() in the Console and help pages will appear in the bottom right panel of R Studio. This includes the starting manual 'An Introduction to R' and links to online help resources.

The quit() or q() commands are used to quit R.

3. Functions

There are many functions included in the standard R download including statistical analysis and graphic displays. By downloading Packages (see below) we can access more specialist functions including those related to the models we study in the Survival Models module. The basic format for a function in R is:

<function name> (<value1>, <value2>, … , <option1>, <option2>, …)

so the function to find logarithm is

log(<value>, base = <base number>) where the base= can be omitted as the log function can only have one value and one option. So log(10000, 10) returns [1] 4 as 10^4 is 10000. If you omit the base option altogether R defaults to natural log (base e) so log(23) returns [1] 3.135494.

The exponential function is exp(<value>) so exp(3.135494) returns [1] 23.

Binomial expansions can be found where ${}^nC_k$ is given by choose(n, k) for numeric values of n and k. choose(162, 91).

With functions it is worth remembering that R is case sensitive so Log Exp Choose will return error messages.

If you need help with a specific <function> then type either help(<function>) or just the <function> name followed by a ?

4. Objects

If we have a variable (often called an Object) X, we can assign a value to it (let's say 2.6) by

X = 2.6

or

X <- 2.6

then print(X) will return the value in the Console. The value will also be stored in the Environment in the top right panel in R Studio. Again, these objects are case sensitive, so X and x are different objects.

We can then do basic mathematics are functions using these objects so with the value above log(X) returns [1] 0.9555114

We can give objects names that are meaningful for the work we are doing rather than just single letters and those names can include _ or . but not spaces. So, we could have Duration or Duration_in_trial or Duration.in.trial as object names instead of the usual algebraic variable t.

Different data types can be stored in Objects:

- numeric (numbers as above)
- characters (sometimes called character strings) We use “ “ or ‘ ‘ to specify the stored object as character data e.g. A = “London”
- logical (TRUE / FALSE the 2 Boolean states) There are then a set of defined logical operators in R which are: == (equal to), != (not equal to), >, <, >=. <=, with & for logical and, | for logical or.

5. Types of Objects

As well as different types of data that can be stored in Objects (see section above), there are different types of (sometimes called classes of) Objects which relate to different data structures:

- Vectors – this is the default type and can contain ordered numbers, characters or logical data. A vector is created with the combine function c(). To create a vector named D with the numbers 42, 53, 66, 68 we use the R command D <- c(42, 53, 66, 68)

  To create a vector E with integers 1 to 5 we can use shorthand E <- c(1:5)

  We can also create a vector from other vectors, F <- c(D, E)

  length(F) will return the number of elements in vector F, class(F) will tell us what type of data is in F and str(F) will give the structure of F and the first few elements.

  There are some other quick ways to create vectors e.g. the sequence command seq(<start value>, <end value>, <increment>) or repeat command rep(<vector>, <n>) which repeats <vector> n times in a new vector.

- Matrices – these are created with the matrix() function specifying a number of rows nrow, and columns ncol.

  matrix(<data created with c() function>, nrow , ncol)

R defaults to filling matrices by column, you need to add byrow=TRUE at the end of the matrix() function to add data by rows. As with vectors, the data type in a matric in R can be numeric, characters or logical. If we need different columns to have a different data type, then use the data frame function instead of matrix()

data.frame(<name1> = <vector1>, <name2> = <vector2>, …) where <name1> is a column name or heading and <vector1> is the vector that populates that column.

If a vector has category values (that is can only take one of a fixed number of values) then we need to use the factor function on that vector in R. Each value that the factor can take is then called a level. We do this by the command as.factor(<vector>)

6. Matrix operations

We can label items in a vector with

names(<vector>) = c("label1", "label2", …)

and in a matrix using colnames() and rownames() =c() in the same way.

Standard arithmetic works with vectors on an element by element basis. So with vectors V and W created we can find V*3, W^2, V+W, log(V). The same applies for a matrix. For proper matrix multiplication rather than the element-by-element operation we need to use %*% instead of *.

To find the sum of and the mean of elements in a row or column of a matrix use rowSums() colSums() rowMeans() colMeans() on the name of the matrix *and remember the capital S and M as R is case sensitive*.

Other matrix functions for a matrix Y are:

determinant	det(Y)
transpose matrix	t(Y)
inverse matrix	solve(Y)

eigen(Y) returns both the eigenvalues and eigenvectors of Y and add $values or $vectors if you only want one of the two outputs.

7. Importing data in R

Data can be imported with packages, via RData files or imported from other files (e.g. csv, Excel). The default place R looks for data files in Windows is a folder called R in MyDocuments. This place is called the Working Directory. You can change the default to a new folder with the function
setwd("<new folder address>")

RData files contain objects created in a previous session of R. You can open them three different ways:

- with the open file icon in the Environment panel in R Studio
- right clicking the file in Windows Explorer and selecting Open With and then R Studio
- inside R with the command load("filename.RData") as long as the file is in the working directory

Data in .csv files can be imported with:

- read.csv("filename.csv") if the file is in the working directory. This function defaults to assuming the data has a header row with column names. Using M <- read.csv() will import the .csv file and store it in a Data Frame called M.
- The Import Dataset icon in the Environment panel in R Studio and then click on the first option From Text (base). The Data Frame can then be named in the Import Dataset window.

It is a good idea to use view(<Data Frame>) to check that the data has been imported correctly and assigned to the right columns.

8. Packages

Packages are collections of code, functions and data sets for use in R. There are thousands freely available via www.cran.r-project.org . We will use certain packages in Survival Models (see separate document on these). To use a package, it has to be downloaded from CRAN and then loaded into R Studio. We can install packages direct using R Studio. Select the Packages tab in the lower right panel and then press Install. If you know the name of the package start to type it in the box and then click on the name when it appears. An alternative from the Console is the command install.packages(“package name”)

Next the package needs to be loaded into the workspace so that it can be used. This can either be done by ticking the box next to the package name in the Package tab or by the Console command library(“package name”).

Help is available on an installed package either by clicking on the name of that package in the Package window or via the command library(help = PackageName)